SELF-PLAY LEARNING STRATEGIES FOR RESOURCE ASSIGNMENT IN OPEN-RAN NETWORKS

A PREPRINT

Xiaoyang Wang
University of Bristol

Jonathan D. Thomas
University of Bristol

Robert J. Piechocki
University of Bristol
The Alan Turing Institute
{xiaoyang.wang, jonathan.david.thomas, r.j.piechocki}@bristol.ac.uk

Shipra Kapoor
BT
shipra.kapoor@bt.com

Raúl Santos-Rodríguez
University of Bristol
enrsr@bristol.ac.uk

Arjun Parekh
BT
arjun.parekh@bt.com

March 5, 2021

ABSTRACT

Open Radio Access Network (ORAN) is being developed with an aim to democratise access and lower the cost of future mobile data networks, supporting network services with various QoS requirements, such as massive IoT and URLLC. In ORAN, network functionality is dis-aggregated into remote units (RUs), distributed units (DUs) and central units (CUs), which allows flexible software on Commercial-Off-The-Shelf (COTS) deployments. Furthermore, the mapping of variable RU requirements to local mobile edge computing centres for future centralized processing would significantly reduce the power consumption in cellular networks. In this paper, we study the RU-DU resource assignment problem in an ORAN system, modelled as a 2D bin packing problem. A deep reinforcement learning-based self-play approach is proposed to achieve efficient RU-DU resource management, with AlphaGo Zero inspired neural Monte-Carlo Tree Search (MCTS). Experiments on representative 2D bin packing environment and real sites data show that the self-play learning strategy achieves intelligent RU-DU resource assignment for different network conditions.

***K*eywords** Open-RAN · Deep reinforcement learning · Self-play · Resource assignment

1 Introduction

Next generation networks promise always-on connectivity everywhere, ultra-low latency and massive capacity. It is seen as a key enabler of the future service revolution [1]. Wireless services have evolved in various directions during the past few decades, with the emerging ecosystem of the Internet of Things (IoT), Ultra Reliable Low Latency Communication (URLLC), high-speed services, etc [2, 3, 4]. To fully utilise the potential of 5G to provide for services with different Quality of Service (QoS) requirements, a versatile structure with reconfigurability is essential. The existing approach of utilising proprietary equipment and vendor-specific design locks mobile network operators (MNOs) into a specific architecture, making upgrade and interoperability of advanced features difficult [5]. Software-oriented and application-adaptive network architecture will be able to serve massive IoT devices and other co-existing network services. Within a 5G Network, the RAN represents significant expenditure, which is estimated to account for 60-65% of the total expense of network ownership [6]. To deliver 5G and beyond services with cost-effective continuous technology upgrades, Open-RAN (ORAN) emerges as the most promising solution.

The concept of ORAN is a vendor-neutral disaggregated network structure which disengages the software from hardware and vendor. This concept was introduced by the 3rd Generation Partnership Project (3GPP) in Release 14 specifications [7]. 3GPP introduced the decomposition of the existing baseband unit (BBU) into three elements, where these are the remote unit (RU), distributed unit (DU) and central unit (CU). In 3GPP release 15 and subsequent

specifications, disaggregation continued in service-based architecture with the split between control and user plane in 5G base stations. A disaggregated network structure has several advantages including: *(i)* The RAN disaggregation brings higher network utilization efficiency [8, 9]; *(ii)* it will enhance network optimisation and improve network quality of experience (QoE) in dynamic environments [10]; and *(iii)* for the RU-DU split, centralized processing across multiple RUs would reduce the cost of baseband resources.

In an ORAN system, the DU is typically hosted in an edge cloud data centre. The connection between DU and RU is one-to-many. Intelligent DU resource allocation for various RU requirements would significantly improve the utilisation of DU resource, facilitating the promised benefits of centralised processing, such as reduced cost and improved user satisfaction. Resource assignment in the RAN has previously been studied along with the development of network centralisation. [11] studies the resource segmentation and allocation problem in Cloud-RAN (C-RAN), considering physical resource allocation and the time resource. The problem is formulated as a stochastic mixed integer nonlinear programming and is solved by successive convex approximation. Heuristics methods are also studied in RAN resource allocation [12, 13]. [14] studies both exact and heuristics methods for the edge computing optimization in C-RAN, including integer linear programming, Matroid-based method and knapsack-based method where they jointly optimize the resource consumption and network latency.

In this paper, we model the RU-DU resource assignment problem in ORAN as a bin packing problem, proposing a self-play reinforcement learning strategy. We apply the combined approach of deep neural network with Monte-Carlo Tree Search (MCTS) [15] for solving this combinatorial optimization problem. With high capacity and function approximation ability, a deep neural network modelled resource assignment policy can generalize across dynamic network conditions. Learning from self-play eliminates the need for demonstrator data, which could be expensive and time-consuming to collect.

The rest of this paper is organised as follows. In Section 2, we review the application of Reinforcement Learning (RL) in next generation networks. Section 3 presents the RU-DU resource assignment problem studied in this paper and the proposed approach. Experiments on the representative bin packing environment and real scenarios are conducted in Section 4. In Section 5, we present the conclusion and future challenges for RL in the ORAN system.

2 Related Work

2.1 RL in Next Generation Networks

Due to its proven capability in intelligent decision making [16, 17, 18], RL has been applied to address communications and networking issues including traffic routing, data offloading, resource allocation, etc [19]. In this section, we focus on RL for resource allocation in 5G and beyond. [20] proposes a dueling-double DQN approach for large scale user association and resource allocation in heterogeneous cellular networks. In mobile edge computing, RL is widely applied for intelligent edge resource allocation [21, 22]. A variety of services with diverse requirements are supposed to be supported by network slicing in next generation networks, which can also be achieved by RL approaches [23, 24]. Spectrum allocation and sharing approaches are studied in [25, 26], improving the link capacity by RL methods.

2.2 RL for Combinatorial Optimisation

The combinatorial optimization (CO) problem can be viewed as searching for the optimal solution in the feasible region [27]. Typical CO problems include the Travelling Salesman Problem (TSP), Bin Packing Problem (BPP), Vehicle Routing Problem (VRP), etc. In recent years, RL approaches have been designed to solve CO problems, by learning policies to either incrementally build CO solutions or iteratively improve solutions [28]. In this section, we especially focus on the neural MCTS method, i.e., the AlphaGo Zero method, reviewing its application in combinatorial optimization problems. AlphaGo Zero is a model-based method, in which the model is given for the agent to plan ahead through MCTS. It belongs to the paradigm of joint approaches, where the decision for CO problems is made by RL policy guided MCTS search. [29] proposes to use a reward-shaping mechanism called ranked reward, enabling single-player games to benefit from adversarial games settings. This approach successfully solved 2D and 3D BPP with a reasonable scale. [30] applied the ranked reward neural MCTS approach to the Morpion Solitaire game, resulting in the best non-human performance. In [31], the CO problems are transformed into the Zermelo game domain, from which optimal solutions can be learned using neural MCTS. CO problem on graphs is studied in [32], in which a graph isomorphism network with MCTS is proposed without human knowledge.

3 Methodology

In this section, we introduce the RU-DU resource assignment in an ORAN system as a variation of the classic bin packing problem. We formulate a 2D bin packing problem to pack items into a single bin while minimizing one dimension of the bin. A self-play reinforcement learning approach is proposed to solve the 2D bin packing problem.

3.1 RU-DU Resource Assignment in ORAN

We consider the edge-central processing model in Fig. 1. The DU is colocated with multiple RUs, connected through fronthaul interfaces. According to the 5G New Radio (NR) functional split design, RUs are small and cost-effective, while DUs are designed to undertake the role of computing centres, conducting the majority of data processing. Resource-intense jobs are centralised at DUs [33]. Intelligent resource assignment between DU and RU will result in high utilisation of DU resources while satisfying latency requirements, bringing the benefit of cost-efficient network operations.

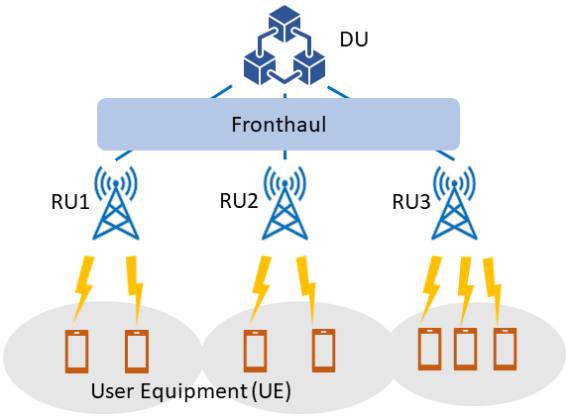


Figure 1: RU-DU architecture in ORAN.

During network operation, the i^{th} RU, represented as R_i, requires compute resource for radio signal processing. R_i raises requests to the DU for resources with capacity c_i and an estimated processing time t_i. These requests are driven by its connection status with UEs, as well as the UE demands. DUs, as mobile edge computing centres, are modelled as resource clusters with capacity C. RU requests are allocated and processed on a batch basis. Note that in this work, we assume the DU-RU connection status is known. We only focus on the resource assignment in established one-to-many DU-RU connections. This problem can be modeled as a 2D bin packing problem (BPP), where RU requirements are modelled as a batch of rectangular items $\mathcal{I} = \{w_i, h_i\}_{i=1}^{N}$ and DU is modelled as a bin of size $(\tilde{W}, \tilde{H})$. Here $w_i = t_i$, $h_i = c_i$, and N is the number of RUs connected to one DU. For the bin (the DU), $\tilde{W}$ is the time limit for processing the current batch of RU requirements, and $\tilde{H}$ is the resource capacity. RU-DU resource assignment is to find the placement of items in the bin. We define (x_i, y_i) as the left-bottom coordinate of a placed item i, with $(0, 0)$ as the left-bottom coordinate of the bin.

Since the cost of DU cluster construction and maintenance is proportional to the amount of resource it has/uses over the operating time, network operators are looking for intelligent assignment strategies where RUs use the minimum resources at DUs with latency requirements satisfied. Minimising the resource assigned for RUs in a DU will result in free physical resources. They can either be set to sleep mode to save energy, or to support other types of services and open interface operations. Given the latency requirement for a batch of RU requirements W^*, the objective is to find a bin with minimal height where all items can be packed into. This can be formalized as the following optimisation problem [1]:

[1]The objective is to minimize the height of the bin since it represents the physical resource. Switching the meaning of height and width of the bin does not affect the results in this paper.

$$\min \tilde{H} \tag{1}$$

$$\text{s.t. } \tilde{W} = W^* \tag{2}$$

$$\beta_{i,j} + \gamma_{i,j} = 1 \tag{3}$$

$$0 \leq x_i \leq \tilde{W} - w_i \qquad ,\forall i \in [1, N] \tag{4}$$

$$0 \leq y_i \leq \tilde{H} - h_i \tag{5}$$

$$x_i - x_j + \tilde{W} \cdot \beta_{i,j} \leq \tilde{W} - w_i \tag{6}$$

$$y_i - y_j + \tilde{H} \cdot \gamma_{i,j} \leq \tilde{H} - h_i \tag{7}$$

Here, $\beta_{i,j}$ and $\gamma_{i,j}$ are binary values. $\beta_{i,j} = 1$ if $x_i \leq x_j$, $\gamma_{i,j} = 1$ if $y_i \leq y_j$. (4) and (5) are to assure items are placed inside the bin. (6) and (7) guarantee there is no overlap between items, due to the nature of resource and time occupation. In addition, we introduce three placement rules on the 2D BPP:

- Items are non-rotatable, as axes are not interchangeable.
- An item can only be placed at the edge of the bin or adjacent to another item in both dimensions. This reduces the size of searching space in this problem.
- If not specified otherwise, the resource occupancy does not need to be contiguous for each item, as long as the required amount of resources are reserved for the same time period.
- Once placed, an item is not allowed to be removed from the bin or to conduct any kind of re-placement. This is to ensure the item placement process is time-finite, helping to find feasible solutions.

Note that in this work we only consider a single-dimensional resource in mobile edge computing, which already brings benefits to ORAN systems. The challenge of multi-dimensional resource assignment is discussed in Section 5.

3.2 Markov Decision Process

We formulate the 2D BPP as a finite Markov decision process (MDP). The bin and items are represented as 2D binary occupancy planes with identical size. For the bin plane, it shows the current placement status. For item planes, unpacked items are placed at the bottom left of the plane while packed items are removed from corresponding planes. The bin plane and item planes are stacked together to form the state of an MDP (see Fig. 2). With one item to be placed into the bin at each step, $\{i, x_i, y_i\}_t,\ t = [1, ..., N]$ encodes the placement location of item i at step t. Considering the possibly non-contiguous resource assignment for each item, $y_i = [y^1, y^2, ..., y^{h_i}]$ is a set of h_i locations rather than a single location (see Fig. 3). For simplicity, we define action $a_t \in \mathcal{A}$ as a tuple $a_t :=< i, x_i >_t$. y_i is then determined by a simple heuristic method introduced in Section 3.3. This process is Markovian. The reward is defined as:

$$r = \begin{cases} \frac{H^*}{\tilde{H}} & \text{if } s_t = s^* \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

Here s^* is the terminal state of the 2D BPP with all items placed in the bin. $H^* = \max(\frac{\sum_{i=1}^{N} w_i * h_i}{W^*}, \max_i h_i)$ is the possible minimal height of the bin. $\tilde{W}$ and $\tilde{H}$ are the width and height of the bin at the terminal state. Thus, $r \in (0, 1]$.

3.3 Ranked Reward with Self-play Reinforcement Learning

As described in Section 3.1, RU requests are processed in batches. Each batch of RU requests forms an instance of the 2D BPP with N items of various shapes. For RU-DU resource assignment, the aim is to learn a generalised policy $\pi(\cdot|s)$ over various RU requirements. Consider the number of possible placement for items, searching for the sparse feasible solutions in large action space brings challenges to policy learning. In addition, the 2D BPP policy requires precise lookahead, since items cannot be withdrawn once placed.

Following the recent success of AlphaGo Zero [15] and Expert Iteration [34], we apply the policy iteration and Monte-Carlo Tree Search (MCTS) on 2D BPP. A two-headed neural network $(\pi_\theta, v_\theta) = f_\theta(s)$ takes the state representation as an input, with the output as a probability distribution over actions $\pi_\theta(a|s)$ and a state-value estimation $v_\theta(s)$. For each state s, M MCTS simulations are performed, guided by f_θ. The outputs of MCTS simulations are refined action probability distribution $\hat{\pi}(a|s)$ and state-value estimations $\hat{v}(s)$. Taking the outputs of MCTS simulations as target policy and target state-value, f_θ is trained in a supervised manner, minimising the following loss function:

$$l = (v_\theta(s) - \hat{v}(s))^2 - \hat{\pi}(a|s) \log \pi_\theta(a|s) \tag{9}$$

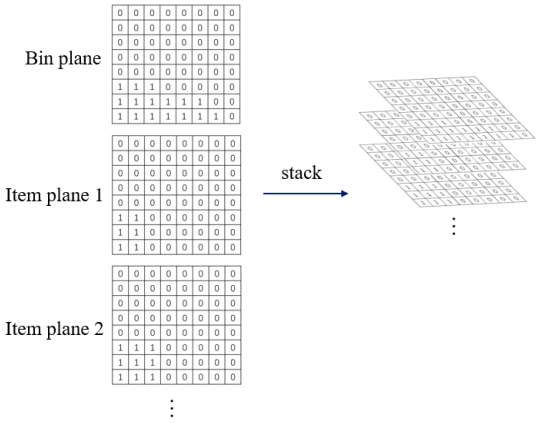


Figure 2: State representation of 2D BPP.

(a) Contiguous resource assignment.

(b) Non-contiguous resource assignment.

Figure 3: Item placement for contiguous and non-contiguous resource assignment. For contiguous resource assignment, the placement of the *blue* item is $\{x_1, y_1\}$, which is its bottom-left coordination. For non-contiguous resource assignment, the *blue* item can be sliced horizontally to two smaller items, placed in $\{x_1, y_1\}$ and $\{x_1, y_2\}$, respectively. Thus y is a vector rather than a single value. In this paper, we apply the non-contiguous resource assignment principle, although not all items will be sliced to smaller ones. For consistency, the y-axis placement of item i with size w_i, h_i is written as $y_i = [y^1, y^2, ..., y^{h_i}]$.

We apply Ranked Reward (R2) as a reward-reshaping method to improve the MCTS search by self-play, presenting an adversary to the current agent from its own past performance [29]. R2 maintains a fixed-length reward buffer $\mathcal{B}$ with recent MDP final rewards. A threshold r_α is used for reward-reshaping, which is the α^{th} percentile of the reward buffer. As presented in [29], the MDP reward r is reshaped to ranked reward z as follows:

$$z = \begin{cases} 1 & \text{if } r > r_\alpha \text{ or } r = r_{\max} \\ -1 & \text{if } r < r_\alpha \\ \textit{random}(1, -1) & \text{if } r = r_\alpha \text{ and } r < r_{\max} \end{cases} \tag{10}$$

Here $\textit{random}(a, b)$ performs a random selection between a and b with equal probabilities. $r_{\max}$ is the upper bound of MDP reward, i.e., 1.0 in the 2D BPP. Algorithm 1 shows the 2D BPP algorithm with ranked reward inspired self-play. Note that in this problem, rewards in $\mathcal{B}$ are from different instances of 2D BPP, which prevents the policy from overfitting to some of the instances. After each training iteration, the sample buffer $\mathcal{D}$ is cleared to stabilize the training process, because new rewards have been added to $\mathcal{B}$, resulting in the change of α^{th} percentile.

4 Experiments and Results

In this section, we first introduce the neural network architecture used in the 2D BPP. To validate the self-play learning approach, we create a representative 2D bin packing environment with 10 items, in which instances are generated by

Algorithm 1: 2D BPP with ranked reward inspired self-play

Input Bin packing problem parameters; a percentile α
Output Trained neural network f_θ
Initialize neural network parameters θ
Initialize sample buffer $\mathcal{D}$ and reward buffer $\mathcal{B}$ and $\mathcal{B}'$. Set $\mathcal{B}' = \mathcal{B}$
for *training iteration* = $1, ..., K$ **do**
  for $episode = 1, ..., J$ **do**
    Generate a new instance of 2D BPP, with items $\mathcal{I}$
    Initialize the bin as a zero occupancy plane
    for *step* = $1, ..., N$ **do**
      Perform a Monte-Carlo Tree Search guided by f_θ
      Sample a_t from MCTS-improved policy $\pi(\cdot|s_t)$
      Get y_t using Algorithm 2
      Perform action <a_t, y_t>, get the next state s_{t+1} and reward r_t
    end
    Save r_N in $\mathcal{B}'$
    Get ranked reward z according to Eq. 10 using $\mathcal{B}$
    Store $(s_t, \pi(\cdot|s_t), z)$ for all steps in $\mathcal{D}$
  end
  for *step* = $0, ..., \tau$ **do**
    Sample mini-batch d from $\mathcal{D}$
    Update θ by minimising Eq. 9 on d
  end
  Clear $\mathcal{D}$
  $\mathcal{B} = \mathcal{B}'$
end

Algorithm 2: Physical resource allocation for action a_t

Input $a_t =< \mathcal{I}_t, x_t >$; $I_t = (w, h)$; Bin occupancy state S^{bin}.
Output y_t
Initialize $y_t = \emptyset$; $i = 0$; $y' = 0$
while $i < h$ **do**
  for y=$y' + 1, y' + 2, ...$ **do**
    if $\sum S^{bin}[x_t : x_t + w - 1, y] = 0$ **then**
      $y_t = y_t \cup [y]$
      $i = i + 1$
      $y' = y$
      break
    else
      continue
    end
  end
end

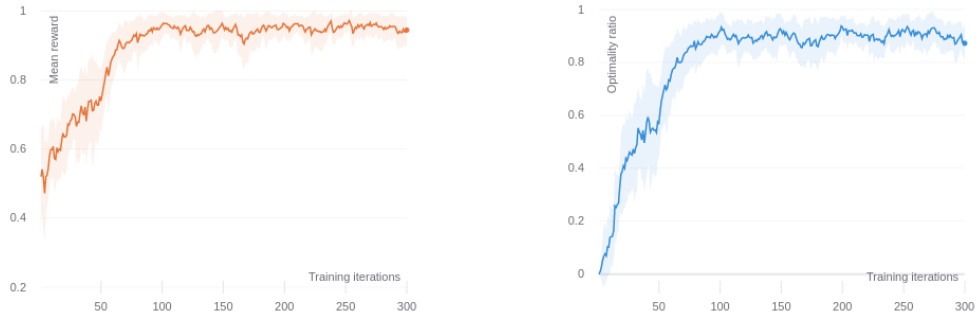
randomly slicing the bin. A policy f_θ is trained on the representative 2D bin packing environment. We then apply the trained policy to a dataset derived from a live network covering the central area of Bristol, UK, to optimise the RU-DU resource assignment.

4.1 Neural Network Architecture

The objective of the 2D BPP in this work is to minimise the height of the bin without setting an initial limitation on the bin size. To make the neural network model applicable, we set "virtual" bin width W' and height H'. Apparently, $W' = W^*$, defined by the latency requirement. The neural network takes as input a $H' \times W' \times (N + 1)$ dimensional tensor, where N is the number of items. We apply the neural network structure as presented in [35], with 15 convolutional layers. For the policy head, a softmax fully connected layer is applied to produce a probabilities distribution over all possible actions, while the value head applies another fully connect layer to estimate the continuous state-value.

4.2 Model Training

In each iteration, 20 different instances are generated from the representative 2D bin packing environment, by slicing a bin with size W^*, H^* into 10 items. Here we set $W', H' = 15$, $W^* = 15$, and H^* can be any integer between *2* to *15*, to train a policy applicable for different RU requirements. Note that we only train one policy f_θ for varying values of H^*. Self-play results for all instances are stored in $\mathcal{B}$ with length 100. At every self-play step, 200 MCTS simulations are conducted to generate improved policy and state-value estimation. For the ranked reward, we set $\alpha = 75$ as suggested in [29]. Fig. 4 shows the mean reward and the ratio of "optimal" packing during training, i.e., the ratio of bin packing results where $\tilde{H} = H^*$.



(a) Mean reward. (b) Optimality ratio.

Figure 4: Mean rewards and optimality ratio of 2D BPP during 300 training iterations.

We compare the self-play learning strategy with a set of baselines: a heuristic virtual resource allocation algorithm (HVRAA) proposed in [36]; the Lego heuristics for bin packing [37]; and the vanilla MCTS using Monte-Carlo rollout. The test is conducted on 100 instances of 10-item 2D BPP, with randomly sampled H^* from $U(2, 15)$ and randomly generated item shapes. Table 1 shows the average reward $\bar{r}$, standard deviation of rewards σ_r and the optimality ratio $R_{\tilde{H}=H^*}$ on the test set. The proposed self-play learning method outperforms the baseline methods on the average reward, with the second lowest standard deviation. The optimality ratio of the self-play learning approach reaches 94%, which is 29% higher than the second highest value. This proves the effectiveness and the robustness of the trained model across various bin packing problems, with items of different sizes. Fig. 5 visualizes three instances of 2D BPP solutions by baseline methods and the proposed self-play learning method, where items are initially generated by slicing bins with $H^* = [7, 12, 5]$, respectively.

	$\bar{r}$	σ_r	$R_{\tilde{H}=H^*}$
HVRAA [36]	0.896	0.239	0.65
Lego heuristics [37]	0.737	0.349	0.44
MCTS	0.936	**0.097**	0.62
Self-play learning method	**0.964**	0.160	**0.94**

Table 1: Performance on 2D BPP for the proposed self-play learning method and other baseline methods.

4.3 Experiments on Real Sites Data

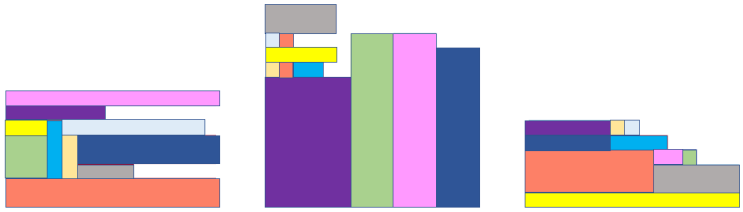
We select a 14×13 km^2 area in Bristol, UK as the potential ORAN deployment area which covers the city centre. It contains more than 100 "4G" sites, currently hosting a mix of 4G and 5G base stations, and more than 10 viable locations for edge computing centres. Fig. 6 shows the map of this area. Sites and potential edge computing centre data is provided by BT[2]. In the ORAN structure, current sites can be seen as RUs, while DUs are allocated in edge computing centres[3]. In this work, we perform RU-DU resource assignment on 2 DUs, located at two different postcode areas. For the physical resource, we focus on the CPU.

The latency between RU and DU can be calculated as

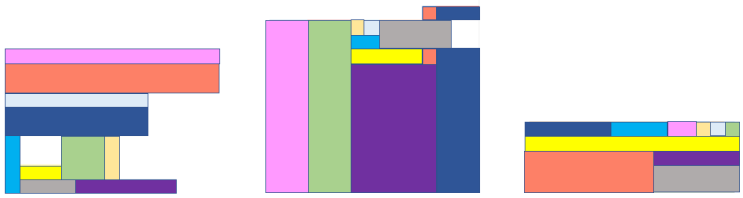
$$t = F \times \frac{d}{c} \tag{11}$$

[2]https://www.bt.com/

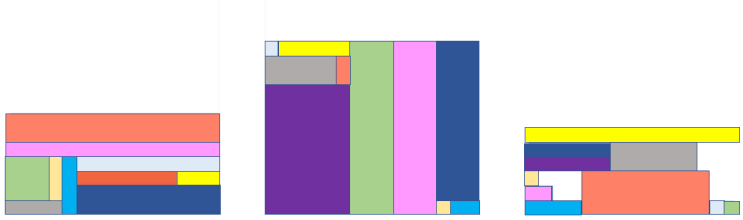
[3]This work is looking at future RAN structure. We use viable locations for edge computing centres as potential DUs in this work. The feasibility of hosting DUs in these locations has not yet been validated by the mobile network operator.



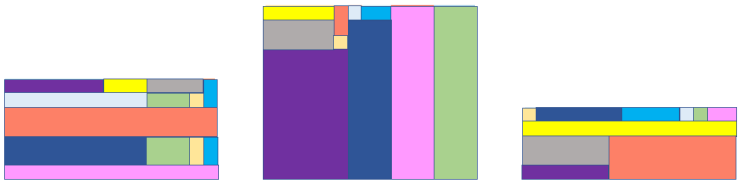
(a) Results of the HVRAA method.



(b) Results of the Lego heuristic method.



(c) Results of the MCTS method.



(d) Results of the self-play learning method.

Figure 5: Results of three 2D BPP instances using baseline methods and the proposed self-play learning strategy. For each instance, the same colour represents the same item.

where d is the straight line distance between the DU and RU, c is the speed of light, and F is a reduction factor, due to the speed reduction in the fibre and the fact that fibre cables are not straight. Empirically, we set $F = 2.5$. The latency requirement for Ethernet mobile fronthaul is $100\,\mu$s [38]. For each DU, it is assumed to be connected with 10 nearest RUs through fronthaul network, which guarantees low latency for all RU-DU connections.

Fig. 7 shows the hourly average CPU requirements for 5 RUs on 4th June 2019, randomly selected from the Bristol map. Peak CPU requirements always occur at around 17:00 and the off-peak hours are from 21:00 to 4:00. We assume the CPU resource requirements are integers sampled from a uniform distribution $\mathcal{U}(\mu - \delta, \mu + \delta)$, where μ and $\mu + \delta$ are the mean and max CPU requirements for one site. Required processing time units are sampled from $\mathcal{U}(1, T')$, where T' is the maximal allowed processing time.

In this work, we focus on peak hours. Different from the representative 2D BPP, there is no guarantee on the existence of a bin (W^*, H^*) whose size equals to the total area of items, i.e., for the reward, $r_{\max} = 1$ does not always hold for real sites data. We introduce a new metric, the resource utilisation, for evaluating the resource assignment performance.

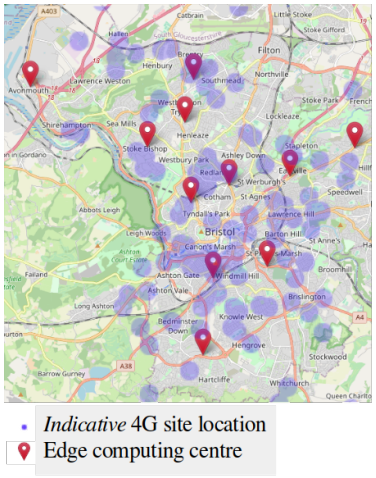


Figure 6: Potential deployment area for ORAN in Bristol, UK.

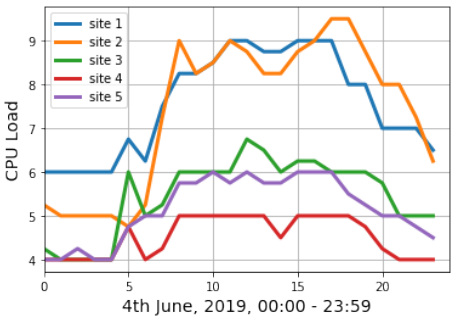


Figure 7: The 24-hour mean CPU load of five 4G sites in Bristol, UK.

The resource utilisation is defined as

$$U = \frac{\sum_{i=1}^{N} w_i * h_i}{\tilde{W} * \tilde{H}} \tag{12}$$

For each DU, we test on 10 instances of peak hour RU requirements. Table 2 shows the average reward $\bar{r}$ and the average utilisation $\bar{U}$ of DU1 and DU2, and Fig. 8 visualises two instances. Note that the optimized bin sizes are marked by dashed lines in Fig. 8. From Table 2 we see that the proposed self-play learning method has the highest average reward for both DUs, with the average utilisation around 86%. Although the neural network model is trained on 2D BPPs with the guaranteed existence of $r_{\max} = 1$, it can be applied on instances that are different from the representative 2D BPPs, achieving the best possible packing solution with minimal bin height. This experiment shows that the model trained in an offline manner can be utilised for dynamic resource assignment, without the need for online interactions or demonstrator data. Using offline training to replace online interaction can avoid querying the live network, reducing the time and query signal cost in network operation. Model re-use can greatly lower the training time and resource cost for network management, as well as reducing the potential operational latency.

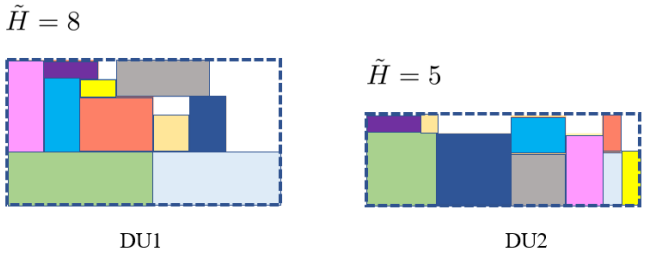
	DU1		DU2	
	$\bar{r}$	$\bar{U}$	$\bar{r}$	$\bar{U}$
HVRAA	0.879	0.809	0.938	0.810
Lego heuristics	0.841	0.774	0.811	0.700
MCTS	0.847	0.781	0.847	0.732
Self-play learning method	**0.943**	**0.869**	**1.000**	**0.864**

Table 2: Resource assignment performance on real sites.

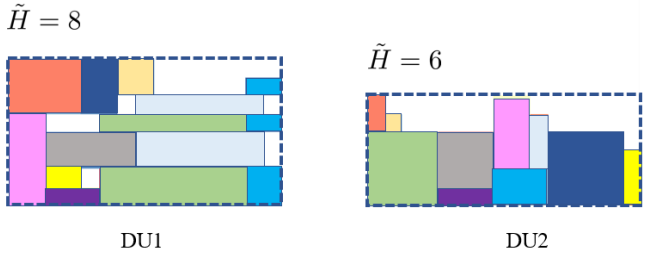
5 Conclusion and Future Work

This work studied the RU-DU resource assignment problem under the ORAN dis-aggregated structure. RU requirements are modelled as 2D items, with the objective of finding a bin with minimal height to pack all items, given a restriction on the bin width. A self-play reinforcement learning strategy is applied to the 2D BPP. The use of ranked reward enables the RL agent to compete against itself, forming a self-play learning structure. Trained on representative BPP environments where the existence of optimal solutions is guaranteed, the model can be applied to bin packing problems outside the original assumption. This would help maintain compact utilisation of DU resources, achieving cost-effective RAN operations. The offline training and well-generalize model reduce both training resource cost and latency for network operations.

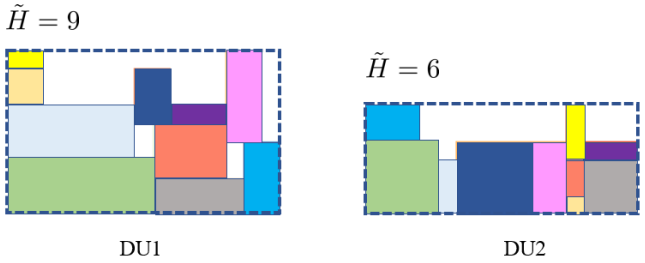
The RU-DU resource assignment with multiple resources remains an open problem. DUs have different capacities for various resources types. One potential way to tackle this problem is to model the multi-resource assignment as a multi-agent RL problem, with each agent dedicated to one resource type. In multi-agent RL, multiple agents can be



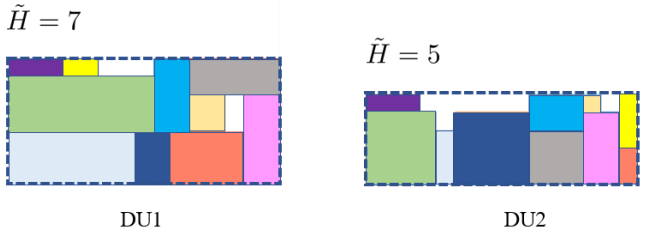
(a) Results of the HVRAA method.



(b) Results of the Lego heuristics method.



(c) Results of the MCTS method.



(d) Results of the self-play learning method.

Figure 8: RU-DU resource assignment results of two instances for DU1 and DU2. Dashed lines show optimized bins $(\tilde{W}, \tilde{H})$ of each problem. Same with Fig. 5, for each instance, the same colour represents the same item.

trained to work in a cooperative way to assign DU resources for RU requirements. Not only is this challenging to RL but also one of the key issues to deploy such algorithms in ORAN.

Another challenge is brought by the scale of RAN. The densification of network leads to large-scale RU-DU distribution. Consider the 2D BPP model presented in this paper, the increasing size of action space will become a bottleneck with a large scale RAN. Besides, as the model complicates or the model scale enlarges, the cost and latency brought by running trained models on the network could be non-negligible.

Acknowledgement

This work is funded by the Next-Generation Converged Digital Infrastructure (NG-CDI) Project, supported by BT and Engineering and Physical Sciences Research Council (EPSRC), Grant ref. EP/R004935/1.

References

[1] Deploying 5G networks, 2020, Nokia corporation. Available at: https://www.nokia.com/networks/5g/mobile/5g-resources/.

[2] Sameer Kumar Singh, Rohit Singh, and Brijesh Kumbhani. The evolution of radio access network towards open-ran: Challenges and opportunities. In *2020 IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pages 1–6. IEEE, 2020.

[3] Hong Ren, Cunhua Pan, Yansha Deng, Maged Elkashlan, and Arumugam Nallanathan. Resource allocation for secure urllc in mission-critical iot scenarios. *IEEE Transactions on Communications*, 68(9):5793–5807, 2020.

[4] Fatima Hussain, Syed Ali Hassan, Rasheed Hussain, and Ekram Hossain. Machine learning for resource management in cellular and iot networks: Potentials, current solutions, and open challenges. *IEEE Communications Surveys & Tutorials*, 22(2):1251–1275, 2020.

[5] Open RAN - The open road to 5G, 2017, SAMSUNG. Available at: https://image-us.samsung.com/Samsungus/samsungbusiness/pdfs/Open-RAN-The-Open-Road-to-5G.pdf.

[6] O-RAN: Towards an Open and Smart RAN, O-RAN Alliance White Paper, October 2018, O-RAN Alliance. Avaiable at: https://static1.squarespace.com/static/5ad774cce74940d7115044b0/t/5bc79b371905f4197055e8c6/1539808057078/O-RAN+WP+FInal+181017.pdf.

[7] Technical specification group radio access network: Study on new radio access technology: Radio access architecture and interfaces (release 14), 3GPP TR 38.801 v14.0.0, 2017, March 2017.

[8] Bo Yi, Xingwei Wang, Keqin Li, Min Huang, et al. A comprehensive survey of network function virtualization. *Computer Networks*, 133:212–262, 2018.

[9] Jun Wu, Zhifeng Zhang, Yu Hong, and Yonggang Wen. Cloud radio access network (C-RAN): a primer. *IEEE Network*, 29(1):35–41, 2015.

[10] Parallel wireless creates OpenRAN "All G" radio access network architecture, June 2020, Parallel wireless white paper. Availabel at: https://www.parallelwireless.com/wp-content/uploads/parallel-wireless-creates-openran-all-g-radio-access-network.pdf.

[11] Jianhua Tang, Tony QS Quek, Tsung-Hui Chang, and Byonghyo Shim. Systematic resource allocation in cloud RAN with caching as a service under two timescales. *IEEE Transactions on Communications*, 67(11):7755–7770, 2019.

[12] Song Yang, Nan He, Fan Li, Stojan Trajanovski, Xu Chen, Yu Wang, and Xiaoming Fu. Survivable task allocation in cloud radio access networks with mobile edge computing. *IEEE Internet of Things Journal*, 2020.

[13] Wenchao Xia, Tony QS Quek, Jun Zhang, Shi Jin, and Hongbo Zhu. Programmable hierarchical C-RAN: From task scheduling to resource allocation. *IEEE Transactions on Wireless Communications*, 18(3):2003–2016, 2019.

[14] Niezi Mharsi. *Cloud-Radio Access Networks: design, optimization and algorithms*. PhD thesis, 2019.

[15] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of Go without human knowledge. *nature*, 550(7676):354–359, 2017.

[16] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*, 2017.

[17] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[18] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[19] Nguyen Cong Luong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim. Applications of deep reinforcement learning in communications and networking: A survey. *IEEE Communications Surveys & Tutorials*, 21(4):3133–3174, 2019.

[20] Nan Zhao, Ying-Chang Liang, Dusit Niyato, Yiyang Pei, and Yunhao Jiang. Deep reinforcement learning for user association and resource allocation in heterogeneous networks. In *2018 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2018.

[21] Yi Liu, Huimin Yu, Shengli Xie, and Yan Zhang. Deep reinforcement learning for offloading and resource allocation in vehicle edge computing and networks. *IEEE Transactions on Vehicular Technology*, 68(11):11158–11168, 2019.

[22] Jiadai Wang, Lei Zhao, Jiajia Liu, and Nei Kato. Smart resource allocation for mobile edge computing: A deep reinforcement learning approach. *IEEE Transactions on emerging topics in computing*, 2019.

[23] Chen Qi, Yuxiu Hua, Rongpeng Li, Zhifeng Zhao, and Honggang Zhang. Deep reinforcement learning with discrete normalized advantage functions for resource management in network slicing. *IEEE Communications Letters*, 23(8):1337–1341, 2019.

[24] Haozhe Wang, Yulei Wu, Geyong Min, Jie Xu, and Pengcheng Tang. Data-driven dynamic resource scheduling for network slicing: A deep reinforcement learning approach. *Information Sciences*, 498:106–116, 2019.

[25] Le Liang, Hao Ye, and Geoffrey Ye Li. Spectrum sharing in vehicular networks based on multi-agent reinforcement learning. *IEEE Journal on Selected Areas in Communications*, 37(10):2282–2292, 2019.

[26] Yun Lin, Meiyu Wang, Xianglong Zhou, Guoru Ding, and Shiwen Mao. Dynamic spectrum interaction of UAV flight formation communication with priority: A deep reinforcement learning approach. *IEEE Transactions on Cognitive Communications and Networking*, 2020.

[27] Yoshua Bengio, Andrea Lodi, and Antoine Prouvost. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*, 2020.

[28] Nina Mazyavkina, Sergey Sviridov, Sergei Ivanov, and Evgeny Burnaev. Reinforcement learning for combinatorial optimization: A survey. *arXiv preprint arXiv:2003.03600*, 2020.

[29] Alexandre Laterre, Yunguan Fu, Mohamed Khalil Jabri, Alain-Sam Cohen, David Kas, Karl Hajjar, Torbjorn S Dahl, Amine Kerkeni, and Karim Beguir. Ranked reward: Enabling self-play reinforcement learning for combinatorial optimization. *arXiv preprint arXiv:1807.01672*, 2018.

[30] Hui Wang, Mike Preuss, Michael Emmerich, and Aske Plaat. Tackling Morpion Solitaire with AlphaZero-like ranked reward reinforcement learning. *arXiv preprint arXiv:2006.07970*, 2020.

[31] Ruiyang Xu and Karl Lieberherr. Learning self-game-play agents for combinatorial optimization problems. *arXiv preprint arXiv:1903.03674*, 2019.

[32] Kenshin Abe, Issei Sato, and Masashi Sugiyama. Solving NP-hard problems on graphs by reinforcement learning without domain knowledge. *Simulation*, 1:1–1, 2019.

[33] Line MP Larsen, Aleksandra Checko, and Henrik L Christiansen. A survey of the functional splits proposed for 5G mobile crosshaul networks. *IEEE Communications Surveys & Tutorials*, 21(1):146–172, 2018.

[34] Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems*, pages 5360–5370, 2017.

[35] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Volodymir Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-RL with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

[36] Wei Zhu, Yi Zhuang, and Long Zhang. A three-dimensional virtual resource scheduling method for energy saving in cloud computing. *Future Generation Computer Systems*, 69:66–74, 2017.

[37] Haoyuan Hu, Lu Duan, Xiaodong Zhang, Yinghui Xu, and Jiangwen Wei. A multi-task selected learning approach for solving new type 3d bin packing problem. *arXiv preprint arXiv:1804.06896*, 2018.

[38] AB Ericsson et al. Common public radio interface (CPRI) specification v7.0, 2015.